WEST Search History

DATE: Friday, June 25, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L24	L20 and (switch or switching or swap or swapping)	9
☐	L23	L20 and (switch$ or swap$)	9
☐	L22	L21 and (switch$ or swap$)	56
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L21	L19 and L13	101
☐	L20	L19 and L6	24
☐	L19	L18 or L17 or L16	3884
☐	L18	((712/228 |712/229 |712/232 |712/233 |712/234 |712/235 |712/236 |712/237)!.CCLS.)	1554
☐	L17	((709/106 |709/107 |709/108)!.CCLS.)	0
☐	L16	((717/106 |717/136 |717/137 |717/138 |717/139 |717/140 |717/141 |717/142 |717/143 |717/144 |717/145 |717/146 |717/147 |717/148 |717/149 |717/151 |717/152 |717/153 |717/159 |717/160 |717/161)!.CCLS.)	2377
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L15	built-in near compil$ and (processor or coprocessor)	10
☐	L14	L13 and legacy near (software or program or code)	9
☐	L13	(optimiz$ near2 (co?processor or coprocessor or processor))	1421
		DB=EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L12	L11	207
		DB=JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L11	(optimiz$ near2 (co?processor or coprocessor or processor))	271
☐	L10	(optimiz$ near2 code)and (co?processor or coprocessor)	2
☐	L9	(optimiz$ near2 code)and (co?processor or processor)	106
☐	L8	L7 and (optimiz$ near2 code)	1
☐	L7	legacy near (software or program or code)	85
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L6	L4 and (optimiz$ near2 code)	76
☐	L5	L4 and (optimiz$ near2 code) and coprocessor	2
☐	L4	legacy near (software or program or code)	636
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L3	L2	82

DB=USPT,PGPB; PLUR=YES; OP=ADJ

	L2	L1 and cache	107
☐	L1	(optimiz$ near2 code) and coprocessor	135

END OF SEARCH HISTORY

Hit List

Search Results - Record(s) 1 through 9 of 9 returned.

☐ 1. Document ID: US 20040111710 A1

Using default format because multiple data bases are involved.

L24: Entry 1 of 9   File: PGPB   Jun 10, 2004

PGPUB-DOCUMENT-NUMBER: 20040111710
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040111710 A1

TITLE: Hardware/software platform for rapid prototyping of code compression technologies

PUBLICATION-DATE: June 10, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Chakradhar, Srimat	Princeton	NJ	US	
Henkel, Jorg	Princeton	NJ	US	
Jakkula, Venkata	Princeton	NJ	US	
Lekatsas, Haris	Princeton	NJ	US	
Sankaradass, Murugan	Princeton	NJ	US	

US-CL-CURRENT: 717/136; 717/106, 717/159

| Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | In |

☐ 2. Document ID: US 20040044994 A1

L24: Entry 2 of 9   File: PGPB   Mar 4, 2004

PGPUB-DOCUMENT-NUMBER: 20040044994
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040044994 A1

TITLE: Restructuring computer programs

PUBLICATION-DATE: March 4, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Bera, Rajendra K.	Bangalore		IN	

US-CL-CURRENT: 717/136

ABSTRACT:

Existing program is restructured based on a set of tasks that the existing program executes. A set of tasks is used, in conjunction with related test cases, to verify the correct functioning of the restructured program, and also to restructure the program. The restructuring process involves information about: (i) the tasks for which the original computer program is used, (ii) the inputs (including their valid ranges) to be provided to the program to accomplish the tasks, and (iii) the outputs generated by the program, known but unfixed bugs, etc. Legacy computer programs or applications for which reliable documentation is either not available or inadequate is desirably restructured for improved understanding and maintenance.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	In

☐ 3.   Document ID: US 20030110478 A1

L24: Entry 3 of 9   File: PGPB   Jun 12, 2003

PGPUB-DOCUMENT-NUMBER: 20030110478
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030110478 A1

TITLE: Portable run-time code synthesis in a caching dynamic translator

PUBLICATION-DATE: June 12, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Duesterwald, Evelyn	Somerville	MA	US	
Desoli, Giuseppe	Watertown	MA	US	
Bala, Vasanth	Sudbury	MA	US	

US-CL-CURRENT: 717/137

ABSTRACT:

A method of producing a caching dynamic translator with portable run-time code synthesis includes programming hardware independent replacement functions in a high level programming language for the caching dynamic translator, and compiling the hardware independent replacement functions to produce hardware dependent computer executable replacement functions.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	In

☐ 4.   Document ID: US 20020147970 A1

L24: Entry 4 of 9   File: PGPB   Oct 10, 2002

PGPUB-DOCUMENT-NUMBER: 20020147970
PGPUB-FILING-TYPE: new

DOCUMENT-IDENTIFIER: US 20020147970 A1

TITLE: Method and system for optimizing code using an optimizing coprocessor

PUBLICATION-DATE: October 10, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Smith, Jack Robert	South Burlington	VT	US	
Ventrone, Sebastian Theodore	South Burlington	VT	US	

US-CL-CURRENT: 717/140

ABSTRACT:

A data processing system includes a central processing unit (CPU) in communication with a system memory. Within the system memory, there is stored legacy code that does not utilize the full features of the CPU. The data processing system also includes a code-optimizing coprocessor in communication with the CPU and the system memory. Control logic within the code-optimizing coprocessor causes the code-optimizing coprocessor to generate optimized code from the legacy code at the same time the CPU executes the legacy code, such that the optimized code is tailored according to the CPU. After the code-optimizing coprocessor has generated at least some optimized code, the code-optimizing coprocessor causes the CPU to automatically utilize at least some optimized code in lieu of at least some of the legacy code.

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	In

☐ 5. Document ID: US 6732220 B2

L24: Entry 5 of 9   File: USPT   May 4, 2004

US-PAT-NO: 6732220
DOCUMENT-IDENTIFIER: US 6732220 B2

TITLE: Method for emulating hardware features of a foreign architecture in a host operating system environment

DATE-ISSUED: May 4, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Babaian; Boris A.	Moscow			RU
Khvatov; Roman A.	Khimky			RU

US-CL-CURRENT: 711/6; 711/203, 717/136, 717/138

ABSTRACT:

The present invention relates to a computer system adapted to efficiently execute binary translated code. In accordance with the present invention, foreign code is stored in a foreign virtual memory space, translated to acquire binary translated code, which is

stored in a host virtual memory space and then executed. The host computer system isolates each virtual memory configuration into separate processes referred to as a virtual machine while enabling multiple virtual machines to exist simultaneously. Execution may switch from one virtual machine to another merely by switching to a new page table, where each page table describes the memory configuration of a virtual machine. Common system level resources are shared by the virtual machines under the control of a virtual memory manager.

22 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

| Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Ir |

☐ 6. Document ID: US 6718539 B1

L24: Entry 6 of 9   File: USPT   Apr 6, 2004

US-PAT-NO: 6718539
DOCUMENT-IDENTIFIER: US 6718539 B1

TITLE: Interrupt handling mechanism in translator from one instruction set to another

DATE-ISSUED: April 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cohen; Ariel	Cupertino	CA		
Perets; Ronen	Cupertino	CA		
Zemlyak; Boris	Cupertino	CA		

US-CL-CURRENT: 717/136; 714/34

ABSTRACT:

An apparatus comprising a translator circuit and a cache. The translator circuit may be configured to (i) translate one or more first instruction codes of a first instruction set into second instruction codes of a second instruction set, (ii) present the second instruction codes to a processor, and (iii) allow interrupts to the processor to be handled seamlessly.

14 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

| Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Ir |

☐ 7. Document ID: US 6691306 B1

L24: Entry 7 of 9   File: USPT   Feb 10, 2004

US-PAT-NO: 6691306
DOCUMENT-IDENTIFIER: US 6691306 B1

TITLE: Use of limited program space of general purpose processor for unlimited sequence
of translated instructions

DATE-ISSUED: February 10, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cohen; Ariel	Cupertino	CA		
Perets; Ronen	Cupertino	CA		
Zemlyak; Boris	Cupertino	CA		

US-CL-CURRENT: 717/139; 703/26, 711/213, 712/208, 712/209, 717/136, 717/137, 717/140,
717/148

ABSTRACT:

An apparatus comprising a circuit configured to (i) translate one or more instruction
codes of a first instruction set into a sequence of instruction codes of a second
instruction set and (ii) present the sequence of instruction codes of the second
instruction set in response to a predetermined number of addresses.

22 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

| Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Ir |

☐ 8. Document ID: US 6631514 B1

US-PAT-NO: 6631514
DOCUMENT-IDENTIFIER: US 6631514 B1

TITLE: Emulation system that uses dynamic binary translation and permits the safe
speculation of trapping operations

DATE-ISSUED: October 7, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Le; Bich-Cau	San Jose	CA		

US-CL-CURRENT: 717/137; 717/136, 717/151, 717/159

ABSTRACT:

The inventive emulator dynamically translates instructions in code written for a first
architecture into code for a second architecture. The emulator designates various

checkpoints in the original code, and speculatively reorders the placement of the translated <u>code instructions according to optimization</u> procedures. If during the execution of the reordered code, a trap should occur, then the emulator resets the original code to the most recent checkpoint and begins executing the original code sequentially in a line-by-line manner until the section is completed or branched out of. The original code is reset by changing the program counter to the checkpoint, and reversing the effects of each instruction which has been executed subsequent to the checkpoint. Thus, any native instructions which correspond to original instructions which occur sequentially prior to the checkpoint have been executed, and any native instructions which correspond to original instructions which occur sequentially subsequent to the checkpoint have not been executed.

52 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

☐ 9. Document ID: US 6301705 B1

US-PAT-NO: 6301705
DOCUMENT-IDENTIFIER: US 6301705 B1

TITLE: System and method for deferring exceptions generated during speculative execution

DATE-ISSUED: October 9, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Doshi; Gautam B.	Sunnyvale	CA		
Markstein; Peter	Woodside	CA		
Karp; Alan H.	Palo Alto	CA		
Huck; Jerome C.	Palo Alto	CA		
Colon-Bonet; Glenn T.	Fort Collins	CA		
Morrison; Michael	Sunnyvale	CA		

US-CL-CURRENT: <u>717/154</u>; <u>712/208</u>, <u>712/222</u>, <u>712/244</u>, <u>717/161</u>

ABSTRACT:

The present invention is generally directed to a system and method for supporting speculative execution of an instruction set for a central processing unit (CPU) including non-speculative and speculative instructions. In accordance with one aspect of the invention a method includes the steps of evaluating the instructions of the program to determine whether the individual instructions are speculative or non-speculative, and assessing each of the speculative instructions to determine whether it generates an exception. For each of the speculative instructions that generates an exception, the method then encode a deferred exception token (DET) into an unused register value of a register of the CPU. In accordance with another aspect of the invention, a system is provided, which system includes circuitry configured to evaluate the instructions of the instruction set to determine whether the individual instructions are speculative or non-

speculative. The system further includes circuitry configured to assess each of the speculative instructions to determine whether it generates an exception. Finally, the system further includes circuitry configured to encode a deferred exception token (DET) into an unused register value of a register of the (CPU.

17 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

| Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Drawi Desc | Ir |

| Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS |

Terms	Documents
L20 and (switch or switching or swap or swapping)	9

Display Format: |- | Change Format |

Previous Page   Next Page   Go to Doc#